Amendments In The Specification

☞ Page 1 & 2, please **replace** the first full paragraph on page 1 (and ending on page 2) with:

-- This disclosure is related to information appearing in U.S. Patent Application S/N (~~unknown~~09/665,892) entitled ERROR CATCH RAM FOR MEMORY TESTER HAS SDRAM MEMORY SETS CONFIGURABLE FOR SIZE AND SPEED, filed 20 September 2000, and issued on 20 November 2001. ~~for~~For the reasons given below that Patent is hereby expressly incorporated herein by reference. The subject matter of the instant disclosure concerns a portion of the operation of a rather large and complex system for testing semiconductor memories. The memory tester described contains within itself an extensive memory subsystem as a component in the overall paradigm for performing tests. Certain capabilities of that memory subsystem are of interest here, in that they serve as the preferred basis for some of the novel subject matter to be disclosed. For reasons of economy in the product, and abetted by the desire to have large amounts of memory available within the tester, a way was developed to use inexpensive memory (DRAM that is slow when randomly accessed) as a substitute for expensive SRAM that is fast even when randomly addressed. The result, when combined with various other memory subsystem features, is a very complex affair involving multiplexing among Groups and interleaving among Banks, as well as implementing such things as variable word width. On the one hand, the particular features we seek to disclose here could be implemented in a system using only SRAM, with a considerable reduction in complexity. There would be a considerable economic penalty, however, which would likely cause the finished product to be an engineering curiosity instead of a viable commercial technique. We have taken a middle ground in this disclosure, where we do not pretend to make the system entirely out of SRAM, although that would certainly be operable. We include the DRAM technique as a matter of course, but we have suppressed much of the messy detail about the internal operation of that DRAM technique, in favor of a modest description of its basic principles. Even a casual reader will conclude that much interesting material (e.g., the different interleaving and addressing schemes and their connection to the different modes of operation) has been omitted. However, every reader will, upon reflection, appreciate that the techniques and features that we seek to disclose do not fundamentally depend on that omitted material, even though they may, in some cases, be slightly influenced by it. So we have a case

of ragged edges that are peripheral to the main issues of interest. The disclosure incorporated above supplies an abundance of detail concerning the DRAM technique. Those that feel that the instant disclosure raises more issues about the memory subsystem than it answers can go to the incorporated disclosure for those answers. It is for that reason that its existence has been made known by being incorporated herein by reference. --;

A1

☞ Page 3, please **replace** the second full paragraph with:

-- As non-volatile memories become larger, denser and more complex, the testers must be able to handle the increased size and complexity without significantly increasing the time it takes to test them. Memory ~~tester~~testers frequently run continuously, and test time is considered a major factor in the cost of the final part. As memories evolve and improve, the tester must be able to easily accommodate the changes made to the device. Another issue specific to testing non-volatile memories is that repeated writes to cells of the memories can degrade the overall lifetime performance of the part. Non-volatile memory manufacturers have responded to many of the testing issues by building special test modes into the memory devices. These test modes are not used at all by the purchaser of the memory, but may be accessed by the manufacturer to test all or significant portions of the memories in as little time as possible and as efficiently as possible. Some non-volatile memories are also capable of being repaired during the test process. The tester, therefore, should be able to identify: a need for repair; a location of the repair; the type of repair needed; and, must then be able to perform the appropriate repair. Such a repair process requires a tester that is able to detect and isolate a specific nonconforming portion of the memory. In order to take full advantage of the special test modes as well as the repair functions, it is beneficial for a tester to be able to execute a test program that supports conditional branching based upon an expected response from the device. --;

A2

☞ Page 12, please **replace** the third full paragraph with:

-- How to replace SRAM with DRAM in the interior test memory of a memory tester was briefly described above, and is the subject of considerable material below. The technique described herein emphasizes an ECR as a principle example, but is by no means limited to DRAM used as an ECR. It will become abundantly clear that the DRAM Memory Sets can also be used to provide high speed, low cost,

A3

reconfigurable interior test memory that can be used to provide Tag RAM's and buffer memories. That done, it would be desirable if arbitrarily many different instances of all these different uses of interior test memory within a memory tester could be allocated and reconfigured as needed from a central collection of memory, rather than existing as separate pre-configured memory mechanisms. --;

A3

☞ Page 14, please **replace** the first full paragraph with:

-- Finally, for certain classes of testing (more about that below), a portion of interior test memory can be used as a Stimulus Log RAM that operates as an ideal DUT to create (as if by emulation, but equivalent substitution is the actual mechanism) the correct conditions that are to exist in an actual DUT at the conclusion of all, or after some intermediate amount of, testing. The idea is to first get the test program's stream of transmit vectors to occur. This stream is then either: (A) Applied to the Stimulus Log RAM (alone) as if it were being exercised in place of the actual DUT (the Stimulus Log RAM does exactly what an actual good DUT would be expected to do); or, (B) Both (A) happens and the stream of transmit vectors is indeed applied to the actual DUT at the same time. In the case of (A), then when (A) is complete an actual DUT will be tested by again generating and sending that same sequence of transmit vectors to the actual DUT. In any event, after either (A) or (B) the Stimulus Log RAM and the DUT ought to have identical contents. Now the actual part can be read to discover its content, while the expected receive vectors are taken from the Stimulus Log RAM, and the comparison results sent to an ECR, Tag RAM's, etc., as usual. In this way the test program does not have to create or contain within itself the particular receive vectors that are the expected response from the applied stimulus. For those classes of test that are compatible with this approach (of which there are many), the test program is made simpler and easier to write and maintain, as fewer internal variables have to scale with, say, the size of the ~~DUT~~DUT's address space. --;

A4

☞ Pages 22 & 23, please **replace** the second full paragraph (beginning on page 22 and ending on page 23) with:

-- We turn now to a discussion of Figure 2, which is a simplified block diagram expansion of the DUT tester 6 of Figure 1, of which there may be as many as thirty-six. It is sufficient at present to describe only one instance thereof. A glance at Figure 2 will show that it is ~~a~~ fairly well populated with

A5

stuff; especially so for a "simplified" block diagram. Some of what is in the DUT Tester 6 and represented in the block diagram is functionally quite complicated, and is not available in "off the shelf" form. It is appropriate here to make two points. First, the primary purpose of including Figure 2 is to describe the basic properties of an important operational environment within the overall Non-Volatile Memory Test System 1. The invention(s) that are fully described in connection with Figure 3 and subsequent figures will either be expansions of mechanisms set out in the following description of Figure 2, or they will be new mechanisms whose motivational premise is found in Figure 2. Either way, as this is written it is not known exactly which of these is before the reader. The goal at present is to provide a simplified yet informative starting point for numerous different Detailed Descriptions of various Preferred Embodiments, so that each of those can be as concise as is appropriate (as opposed to one "jumbo" Specification that discloses everything about each different invention). The second point is that the expanded or extended material, while in general overall agreement with Figure 2, may contain information that does not "match-up" exactly with the simplified version. This does not mean there has been an error, or that things are fatally inconsistent; it arises because it is sometimes difficult or impossible to simplify something such that it is the exact image in miniature. The situation is rather like maps. A standard size road map of Colorado will show that when going east on I-70 you can go north on I-25 at Denver. It looks like a left turn. And while it did used to be an actual left turn, it isn't one now, and a detailed map of that intersection will show a sequence of component turns and intervening road sections. But no one would say that the standard size road map is wrong; it is correct for its level of abstraction. Similarly, and despite its fairly busy appearance, Figure 2 is indeed a simplification operating at a medium level of abstraction, but some seeming left turns are not simple left turns at all. --;

a5

☞ Page 23, please **replace** the first full paragraph with:

-- As is shown in Figure 1, the major input to the DUT Tester 6 is an instance of the Test Site Bus 5, which originates from a Test Site Controller 4 that is associated with the instance of the DUT Tester 6 that is of interest. The Test Site Bus 5 is coupled to a Micro-Controller Sequencer 19, which may be likened to a special purpose microprocessor. It fetches instructions from a program stored in a program memory, which may be either internal to the Micro-Controller Sequencer 6 (PGM SRAM 20) or external thereto (EXT. DRAM 21). Although these two memories appear to be addressed by what is essentially

a6

a logically common address 63 that serves as a program counter (or, instruction fetch address), and either can be a source of programming to be executed, note that: (1) Only one of the memories performs instruction fetch memory cycles during any period of time; and (2) In fact they are addressed by electrically different signals. The SRAM is fast and allows genuine random access, but consumes valuable space within the Micro-Sequence Controller 19 (which is a large IC), so its size is limited. The external DRAM can be provided in adjustable amounts of considerable quantity, but is fast only when accessed in sequential chunks involving linear execution and no branching. Programming in the SRAM 20 most often is that which is intensely algorithmic, while the EXT. DRAM 21 is best suited for material not readily generated by algorithmic processes, such as initialization routines and random or irregular data.

--;

☞ Page 24, please **replace** the first full paragraph with:

-- The eight sixteen-bit ALU's (24) each have a conventional repertoire of arithmetic instructions built around associated sixteen-bit result registers (each ALU has several other registers, too). Three of these result registers and their associated ALU's are for generating X, Y and Z address components 27 that are variously combined into a complete address to be supplied to the DUT. Two more of the eight ALU/registers (DH & DL) are provided to assist in the algorithmic creation of ~~thirty-two bit~~(thirty-two)-bit data patterns 28 that are divided between a most significant portion (DH) and a least significant portion (DL). A final three ALU/registers (A, B, C) are used as counters and contribute to the production of various PROGRAM CONTROL FLAGS 25 that assist with program control and branching on completion of some programmatically specified number of iterations or other numerical condition. These PROGRAM CONTROL FLAGS 25 are sent back to the Micro-Controller Sequencer 19, where they affect the value of the instruction fetch address in ways familiar to those who understand about microprocessors. There are also various OTHER FLAGS 55 that also can be used to effect program branching. These originate with various ones of the other mechanisms within the DUT Tester 6 that are controlled by the different fields of the fetched instruction word. One specific additional flag is expressly shown as a separate item: VEC_FIFO_FULL 26. In another drawing having somewhat less detail it might be lumped in along with the OTHER FLAGS 55. We have separated it out to assist in explaining one aspect of the operation of the Micro-Controller Sequencer 19. --;

☞ Page 27, please **replace** the third full paragraph with:

-- Circuit 40 can perform three functions: assemble vector components (38, 39) into an ordered logical representation an entire vector that is to be applied (transmitted) to the DUT; apply an arbitrary dynamic correspondence (mapping) between the ordered bits of the logical representation of the transmit vector and the actual physical channel number of the Pin Electronics (i.e., which probe tip) that will contact the DUT on behalf of that signal (i.e., that bit in the vector); and, cooperate with the compiler in the division of an entire logical vector into pieces to be applied separately and in order (serialization) for DUT's that admit of such a thing. Which of these functions is performed is determined by control signals from an SRAM 41, which is also addressed in accordance with a field in the two hundred and eight bit instruction fetched by the Micro-Controller Sequencer 19. The output of Circuit 40 is an up to sixty-four bit vector 44 that is applied to a Vector FIFO 45, which when full generates the signal VEC_FIFO_FULL 26, whose meaning and use was discussed above. The vector at the top of the Vector FIFO 45 is removed therefrom upon receipt of a signal VEC_FIFO_UNLOAD 47 that originates at a Period Generator 49 (to be discussed shortly). Such removed vectors (46) are applied to a Timing / Formatting & Comparison circuit 52 that is connected to the DUT via the associated instance of Pin Electronics 9. That is, each instance of Pin Electronics 9 receives Transmitted & Received Vectors 7 and Pin Electronics configuration information 8 from its associated Timing / Formatting & Comparison circuit 52. --;

A8

☞ Pages 28, please **replace** the second full paragraph with:

-- We turn now to the Period Generator 49 and its associated Timing SRAM 51. These respond to an eight bit signal T_SEL 43 that, for each two hundred and eight bit instruction fetched by the Micro-Controller Sequencer 19, determines a duration for the associated operation of the Timing / Formatting & Comparison circuit 52. T_SEL 43 is member of the Various Control Values & Instructions 42 that are represented by the different fields within the fetched instruction. As an eight bit value it can represent or encode two hundred and fifty-six different things. In this case those "things" are ~~twenty-eight bit~~(twenty-eight)-bit values stored in the Timing SRAM 51 and that are addressed by T_SEL. Each addressed ~~twenty-eight bit~~ (twenty-eight)-bit value (23) specifies a desired duration with a 19.5 picosecond resolution. The sequence of accessed ~~twenty-eight bit~~(twenty-eight)-bit duration values (23) is stored in a Period FIFO

A9

50 so that the individual members of that sequence will be retrieved and applied in synchronism with the retrieval of their intended corresponding vector, which is stored in the Vector FIFO 45. --;

☞ Page 29, please **replace** the second full paragraph with:

-- Refer now to Figure 3, which is a simplified block diagram 64 of the Interior Test Memory 128 in the block diagram of Figure 2. It receives a ~~forty-eight bit~~(forty-eight)-bit mapped address 30 from the Address Mapper 29, which is applied to various Address Classifiers 77, 78 and 79. The Address Classifiers are associated with Memory Sets 73 - 76, which are each complete memory mechanisms that can individually perform various functions, such as being an ECR 32. Two of these Memory Sets (73, 74) are of external DRAM, while two are of internal SRAM. The two external DRAM Memory Sets will always have the same Address Classifier function in effect, and thus share one common Address Classifier 77. The internal SRAM Memory Sets 75 and 76 each have their own associated Address Classifiers, 78 and 79, respectively. These Address Classifiers can either pass an address through unchanged, or modify it in ways to be described in some detail in due course below. --;

☞ Page 36, please **replace** the fourth full paragraph with:

-- Clearly, some of the above arise from the multiplexing and interleaving scheme. The multiplexing and interleaving schemes are, of course, limited to the DRAM Memory Sets (the SRAM Memory Sets go fast to begin with). This does not mean, however, that these same abilities or modes of operation cannot be supported by the SRAM Memory Sets. In general, memory transactions that can be directed to one Memory Set can be directed to any other, subject only to size constraints. An SRAM Memory Set will honor any style of operation that a DRAM Memory Set would. The difference is how the Memory Set controller internally implements the desired transaction. ~~for~~For example, in the case of an Analysis Read (compose) an SRAM Memory Set need not bother beyond doing the simple read, since its data is already composed in the first place. --;

☞ Page 37, please **replace** the first full paragraph with:

-- To resume our discussion, refer now to Figure 6, which is a simplified block diagram 129 of the Address Classifiers (77, 78, 79) shown in Figures 3 and 4. It will be recalled that it is desirable that a plurality of tables located ~~with in~~within a Memory Set (such as Tag RAM's, Buffer Memories, etc.) are all ~~addresses~~addressed by the same range of applied DUT addresses, but that these tables do not overlap. This means that the tables have got to have separate ranges of addresses, even though the sequence of addressed locations therein correspond to each other. One function of an Address Classifier is to shift an address range by some amount so that the existence of the range is preserved, but that its location is changed by the amount of the shift. For performance reasons we do not actually add some arbitrary value to the address with an adder: that would be slow and consume high amounts of space on the die of the VLSI circuit of which the Address Classifier is a part. Instead, we force the upper address bits (for an address to be moved) to have different values. Say, for example, that the range of interest is described by ten least significant bits. Then that range can be relocated by forcing any of the more significant (and otherwise unused) bits to some different value. --;

a12

☞ Page 37, please **replace** the second full paragraph with:

-- In Figure 6 this is accomplished by a collection of MUX's 130a-z (either thirty-two or twenty-one in number, for Address Classifiers 77 or 78/79, respectively), one for each address bit. Each MUX in this collection 130 receives a constant logic zero, a constant logic 1 and the ~~a~~ corresponding actual bit of the address to be classified. The function of each MUX is to select which bit positions in the classified address are forced to ones or zeros, and which are allowed to be driven by their true value. To facilitate this, each MUX in the collection 130 receives a ~~two-bit~~two-bit control input originating at a respective latch 131a-z in a collection of such latches. These latches are set by commands sent via the Ring Bus 85. --;

a13

☞ Page 37, please **replace** the third full paragraph with:

-- Address Classification also cooperates with the upstream MUX's 85-87 of Figure 4, in that those MUX's determine which of the ~~forty-eight bit~~(forty-eight)-bit address bits 30 from the Address

a14

Mapper 29 are passed on as a ~~thirty-two bit~~(thirty-two)-bit address, and what bit position those passed on bits occupy. --;

☞ Pages 38 and 39, please **replace** the third full paragraph (beginning on page 38 and ending on page 39) with:

-- The final step in the data classification process is shown in Figure 10, which is a simplified block diagram 155 of two kinds of masking circuits. The first masking circuit is for forcing data bits in selected positions to have particular values. This portion of Figure 10 includes thirty-two 3:1 MUX's 156a-z, controlled by respectively associated control ~~register~~registers 157a-z that are each setable by the Ring Bus 85. It operates in exactly the same manner as circuit 129 of Figure 6, except that the bits being forced are data bits instead of address bits. The forcing of data bits, it will recalled, is useful when extraneous data bits are to be disregarded by simply declaring them to be "good," for example. The second masking function is not necessarily a part of the data classification function, but is logically located at this point in the data flow, and is itself another masking operation. Since it supports a feature of interest (a read-modify-write for narrow word operation) it has been included here for completeness. MUX's 158a-z choose between the masked data having forced bits and the data read from a location in the Interior Test Memory that is to be modified in part and in part re-written without modification. The selections as to which bits are which is performed by the MUX's 158a-z on a bit-by-bit basis, in accordance with the respective contents of the control latches 159a-z, which get set according to traffic on the Ring bus 85. If a MUX 158 selects a bit from the forced data bits (MUX's 156) then that bit is being modified. If, on the other hand, that bit is coupled to what was read from memory and supplied by the Memory Set Controller, then it is being re-written without modification. In any event, the result is compressed, shifted and masked data 139 that is the output from the data classification process of Figure 7. --; and

☞ Page 39, please **replace** the first full paragraph with:

-- It will be recalled that for certain classes of testing a Stimulus Log RAM operates as an ideal DUT to create the correct conditions that are to exist in an actual DUT at the conclusion of all, or after some intermediate amount of, testing. The idea is to get the test program's stream of transmit vectors applied to the Stimulus Log RAM as if it were being exercised in place of the actual DUT (the Stimulus

Log RAM does exactly what an actual good DUT would be expected to do). That stream of transmit vectors may also be applied to the actual DUT at the same time. In any event, the Stimulus Log RAM and the DUT ought to have identical contents after both have responded to the same sequence of transmit vectors. Now the actual part can be read to discover its content, while the expected receive vectors (compare data) are taken from the Stimulus Log RAM, and the comparison results sent to an ECR, Tag RAM's, etc., as usual. In this way the test program does not have to create or contain within itself the particular receive vectors that are the expected response from the applied stimulus. For those classes of test that are compatible with this approach, the test program is made simpler and easier to write and maintain, as fewer internal variables have to scale with, say, the size of the ~~DUT~~DUT's address space. --.